Jurnal Ilmu Komputer dan Informasi (Journal of Computer Science and Information). 8/1 (2015), 35-44
DOI: http://dx.doi.org/10.21609/jiki.v8i1.282

COMPARATIVE STUDY OF RTOS AND PRIMITIVE INTERRUPT IN EMBEDDED SYSTEM

Dwi M J Purnomo[1], Machmud R Alhamidi[1], Grafika Jati[1], Novian Habibie[1], Benny Hardjono[1,2] and Ari Wibisono[1]

[1]Faculty of Computer Science, Universitas Indonesia, Kampus Baru UI, Depok, 16424, Indonesia
[2]Faculty of Computer Science, Universitas Pelita Harapan, Thamrin Blvd, Lippo Village-Tangerang, 15811, Indonesia

E-mail: ari.w@cs.ui.ac.id

Abstract

Multitasking is one of the most challenging issues in the automation industry which is highly depended on the embedded system. There are two methods to perform multitasking in embedded system: RTOS and primitive interrupt. The main purpose of this research is to compare the performance of RTOS with primitive method while concurrently undertaking multiple tasks. The system, which is able to perform various tasks, has been built to evaluate the performance of both methods. There are four tasks introduced in the system: servo task, sensor task, LED task, and LCD task. The performance of each method is indicated by the success rate of the sensor task detection. Sensor task detection will be compared with the true value which is calculated and measured manually during observation time. Observation time was varied after several iterations and the data of the iteration are recorded for both RTOS and primitive interrupt methods. The results of the conducted experiments have shown that, RTOS is more accurate than interrupt method. However, the data variance of the primitive interrupt method is narrower than RTOS. Therefore, to choose a better method, an optimization is needed to be done and each product has its own standard.

Keywords: *multitasking, RTOS, primitive interrupt, method performance, success rate*

Abstrak

Multitasking adalah salah satu tantangan besar dalam industri otomasi yang sangat bergantung pada *embedded system*. Untuk melakukan *multitasking* pada *embedded system,* terdapat dua metode utama, yaitu RTOS dan *primitive interrupt*. Tujuan utama dari penelitian ini adalah untuk membandingkan kinerja metode RTOS dengan *primitive interrupt* ketika mengerjakan banyak pekerjaan secara bersamaan. Sistem yang mengerjakan beragam pekerjaan dibuat untuk mengevaluasi kinerja dari kedua metode. Terdapat empat pekerjaan yang diberikan kepada sistem, motor servo, sensor ultrasonik, LED, dan LCD. Kinerja dari metode diindikasi oleh keberhasilan sensor ultrasonik untuk mendeteksi objek yang bergerak. Hasil deteksi sensor ultrasonik akan dibandingkan dengan nilai sebenarnya yang diperoleh dari perhitungan dan pengukuran manual selama waktu pengamatan. Waktu pengamatan akan diubah setelah dilakukan iterasi dan data dari setiap iterasi akan dicatat untuk metode RTOS dan metode *primitive interrupt*. Berdasarkan eksperimen yang dilakukan, RTOS lebih akurat apabila dibandingkan dengan metode *primitive interrupt*. Akan tetapi, varian nilai dari *primitive interrupt* lebih sempit dibanding dengan RTOS. Oleh karena itu, untuk menentukan metode yang lebih baik, optimisasi perlu dilakukan karena setiap produk mempunyai standar masing-masing.

Kata Kunci: *multi tasking*, *RTOS, primitive interrupt, kinerja metode, keberhasilan*

1. Introduction

Automation industry has been soaring recently. The labour empowerment has been hampered with various human limitations such as human error which in turn causes inaccuracies, and not to mention inconsistencies in the applied standard of procedure. Labour performances are highly depended on the conditions of environment, human body (fatigue, not feeling well), degree of comfort, etc. These weaknesses are endeavoured to be eradicated by empowering robots as replacements of human labourer.

Robot possess miscellaneous advantageous over human employees. Error would be minimum in robot, the accuracy of robot is undoubtedly high, and finally robot is capable to undertake repetitive task identically (high precision/small variance) [1]. Furthermore, robot performance would not be affected by the altering of the job site conditions.

In the manufacturing industries, robot selection has become challenging issue. There are various criteria underlie robot selection, such as capabilities, specifications, range of applications, and constant time during repetitive tasks [2]. Constant and accurate time during operation is predominant criteria which should be elaborated for robot selection. Inconsistency and inaccurate timing during job undertaking, would lead to error that will be accumulated each time and likely to cause product rejection by means of tolerance incompliance. Therefore, while deciding the automation machine to be chosen, both accuracy and precision are necessitated to be considered.

In this study, Real Time Operating Systems (RTOS) will be evaluated and compared against primitive interrupt for embedded system since both are frequently utilized in industrial robot implementation. RTOS operation is based on the priority which is predetermined by the deadline time [3]. The scheduling will be undertaken by RTOS to meet the deadlines, unlike the primitive interrupt which is prescribed manually. There are many kinds of RTOS, for example free RTOS, Real-Time Unit (RTU) hardware RTOS, the pure software Atalanta RTOS, and a hardware/software RTOS composed of part of Atalanta interfaced to the System on-a-Chip Lock Cache (SoCLC) hardware [4]. On the other hand, there are two categories of primitive interrupt, preemptive and non-preemptive interrupt. In the preemptive interrupt tasks could be halted in the middle of processing and replaced by another task which have higher priority. On the contrary, in non-preemptive interrupt the higher priority task is must wait until the running task finished and all tasks have to cooperate to undertake the entire job.

While RTOS is undertaking multiple tasks, each task is placed into one of these four states, running, ready, block, and suspended state. First of all, if task is in the running states, it means that the processor is being utilized to perform the task. Secondly, a task would be in the ready state if the task is executable because is in neither blocked nor suspended state, and has not been executed because a higher priority task is still in the running state. Thirdly, a task is placed into a blocked state if the task is waiting for triggers. Triggers could be external triggers such as button or switch and temporal trigger (for instance timer delay). Finally, a task would not enter nor exit the suspended state, unless that task is called by a certain code.

Research on RTOS has been conducted for many years. For example, a research which was conducted by Neishaburi et al. (2007) which has focused on robustness improvement of RTOS [5]. Secondly, Maruyama et al. in 2010 have investigated energy conservation by implementing RTOS in hardware [6]. In the robotic field, Liu Xianhua et al. (2003) have conducted a research that implemented Digital Signal Processing (DSP) devices and RTOS platform as a control architec-ture of autonomous robot [7]. While in 2006, Serker et al have developed network-based real-time for robot control system [8]. In 2014, Atmadja et al. examined RTOS in Embedded Linux for mobile robot implementation [9]. Finally, Hoxing Wei et al. (2014) have conducted a research in Robot Operating System (ROS) which utilized the hybrid of RTOS and General Purpose Operating System (GPOS) [10].

In this research, the performance of RTOS will be investigated while undergoing multiple tasks. The results of which will be compared against primitive interrupt performance in the similar scenario.

2. Methods

Components utilized to evaluate the performance of both RTOS and primitive interrupt comprises: DT-Combo AVR-51 Starter Kit which is employed as the main board, microcontroller ATMEL ATmega32, servo motor GWS S03N Standard, SRF 08 ultrasonic sensor, and LCD and LED which are embedded in the main board. The scheme of the architecture of the system is shown in Figure 1. The specifications of the microcontroller used in the experiment are shown in Table 1. Meanwhile, the specifications of the ultrasonic sensor and the mainboard are listed in Table 2 and Table 3, respectively.

In this research, microcontroller would be imposed multitasking job. The performance of the system would be examined while undertaking multiple tasks, especially in time precision and accuracy. There are four concurrent tasks that will be appointed to the system. First of all, the system had to operate and control servo motor motion based on the code. The servo motor will rotate reciprocally from 0^o to 180^o. In addition, there was another microcontroller to control another servo motor. The rotation of the latter servo motor would be converted to be translation by connected it to rack and pinion mechanism. The position of the rack (translation) would be observed by ultrasonic sensor (SRF08) and become the second task of the system. The amount of which the ultrasonic sensor detected the rack would be recorded in a prescribed observation time. The schematic of servo motor and rack and pinion mechanism are shown in Figure 2.

The two last tasks appointed to the system were the counting that would be displayed on LCD and LED which indicated the conformity of

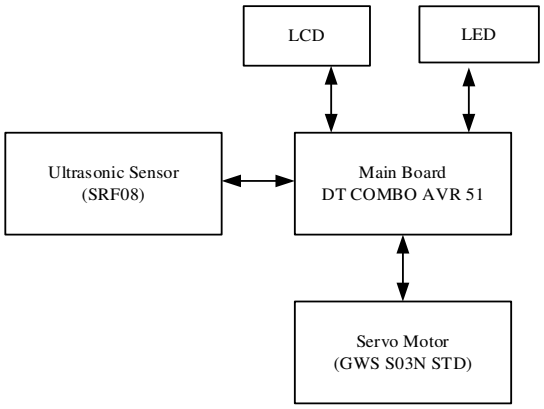


Figure. 1. Scheme of the system's architecture.

TABLE 1
MICROCONTROLLER'S SPECIFICATIONS

Features	Value
In-System Programmable Flash	32 KB
EEPROM	1024 B
SRAM	2 KB
General Purpose I/O lines	32
Voltage	4.5V - 5.5V

TABLE 2
ULTRASONIC SENSOR'S SPECIFICATIONS

Features	Value
Max. distance	11 m
Max. Operating distance	6 m
Voltage	5V

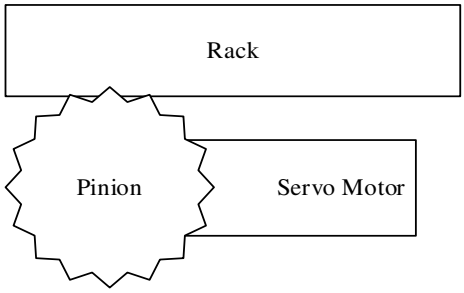


Figure 2. Scheme of the rack and pinion mechanism.

TABLE 3
MAINBOARD'S SPECIFICATIONS

Features	Value
Voltage	9-12 V
LCD	8x2 characters
LED	8 pcs (0-7)
Switch button	8 pcs (0-7)
Path I / O	35 pins
Dimensions	17.5 cm x 12.5 cm x 2 cm

TABLE 4
TASKS DESCRIPTION

Components	Task
LED	Counting per 1 s
LCD	Displaying hit counting
Servo	Reciprocally rotating (0 – 180°)
Ultrasonic sensor	Transmit/receive ultrasonic wave

time measurement. The time recorded will be evaluated using the predetermined time, both accuracy and precision. This scenario also will be appointed to the system with primitive interrupt and thereupon the behaviour of each system would be compared. The tasks that would be executed are summarized in Table IV.

Free RTOS and Non-preemptive Interrupt

FreeRTOS was used in this research due to its various advantageous. The benefits of using RTOS constitute its capability of providing solution for many different architectures, its reliability, undergoing active development, free of charge, its simplicity, and various minor benefit [11].

The task's priorities in the freeRTOS are defined in the FreeRTOSConfig.h. These priorities are ranging from 0 to (`configMAX_PRIORITIES` -1) [12]. The value of (`configMAX_PRIORITIES` -1) could be described to be any number, the only restriction is the RAM capacity. Never-theles, if the value of `configUSE_PORT_OPTIMISED_TASK_SELECTION` is set to be 1 in the `FreeRTOSConfig.h`, the maximum value of `configMAX_PRIORITIES` cannot exceed 32. Meanwhile, the task which has priority of 0 is defined by `tskIDLE_PRIORITY`.

On the other side, non-preemptive interrupt was used in this study. This type of interrupt was used because in the experiment the tasks were deemed have equal priorities. Therefore, to make it comparable with the RTOS which use similar priority on each task, non-preemptive interrupt was used instead of preemptive interrupt.

The priorities of the interrupt are defined in the microcontroller datasheet [13]. For instance, `RESET` which has the highest priority and `TIMER0 COMP` which was used in this research. To define the interrupt, Interrupt Service Routine (ISR) must be included into the code. Further on, to activate the global interrupt so that the interrupt would be running, macro `sei()` was called.

Servo Motor

The architecture of servo motor is shown in Figure 3. There are four foremost components inside servo motor, potentiometer, motor shaft, internal signal generator, and comparator. Poten-tiometer angular position resembles motor shaft position by means of gear power transm-ission. The angu-

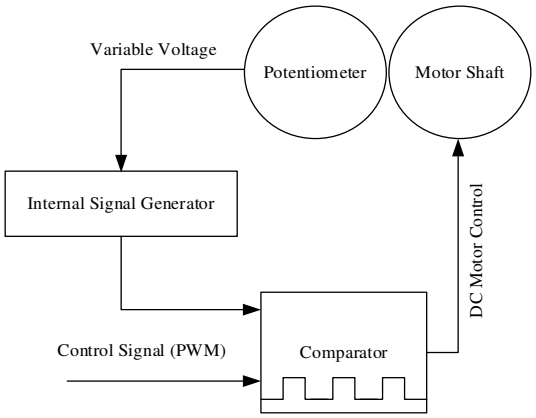


Figure 3. Architecture of servo motor.

lar position of potentiometer defines the variable voltage generated by internal signal generator. The above-mentioned voltage there-upon transmitted to te comparator to be compared with control signal which is Pulse Width Modulated (PWM) signal (Figure 4) .

The difference of the voltage then define the direction of the rotation of motor shaft (whether clockwise or counter clockwise). Motor shaft is rotated by utilizing Direct Current (DC) voltage, thus by the existence of gear mechanism will rotates the potentiometer. The potentiometer's angular position will be altered afterward that lead to the change of the generated voltage from internal signal generator. The aforementioned consecutive events will run continuously until the equivalence of the signals in the comparator attained.

PWM signal is employed to control the servo motor. As shown in Figure 4. There are two parts in PWM signal, high voltage (1) and low voltage (0). The DC voltage generated is underlied by the difference between the generated signal by means of potentiometer and the average of the PWM signal. The average of the PWM signal can be calculated by empowering equa-tion(1). X is the high voltage duration which is ranging from 1 ms to 4 ms, T is PWM signal period (20 ms), and V is voltage. Meanwhile, to define the angular position of the motor shaft by means of the potentiometer, equation(2) can be utilized. X_o is the high voltage duration to be positioned in the smallest angle (in this case 4 ms for 0°). X_f is the high voltage duration to be positioned in the largest angle (in this case 1 ms for 180°). Finally θ_m is the range of the angle (180°).

$$\bar{V} = \frac{X}{T} V \tag{1}$$

$$\theta = \frac{X - X_o}{X_f - X_o} \theta_m \tag{2}$$

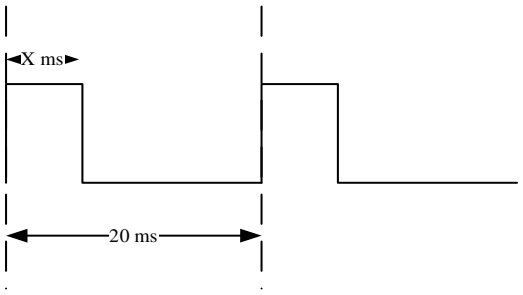


Figure 4. PWM signal.

Ultrasonic Sensor

The ultrasonic sensor works based on the frequency of the signal transferred to the magnetic membran in the actuator. The current will flow into the actuator and will vibrate the magnetic membrane by means of Lorentz force. The vibration frequency will be equal to the excitation signal frequency, hence by determining the excitation signal frequency the vibration frequency can be defined. The vibration of the mag-netic membrane thereupon will lead to the generation of ultrasonic sound that will be fired to the object to measure the distance of which. The distance of the object is estimated by reckoning of the duration of the reflected ultrasonic signal perceived by the reciever (equation(3)). L is the distance of the object, T is the duration of the reflected signal recieved by the reciever, and c is the sound speed (340 m/s).

$$L = \frac{T}{2} c \tag{3}$$

As described in the previous chapter, the ultrasonic sensor employed in this research is SRF 08. The maximum distance can be measured bu utilizing this sensor is 11 m (Table II). The aforementioned value is corresponding to the interval of which the ultrasonic remain idle while shooting ultrasonic wave (65 ms). This value can be altered to reduce the interval time of which the u;trasonic sensor is idling. If the interval time is abridged, the firing repetition will be more frequent. Thus, the delay of which the position of the object overlooked will be minimized, and it will lead to the surging of the accuracy. Equation(4) can be empowered to calculate the maximum distance ultrasonic sensor can detects. L_m is the maximum distance and *reg* is range register which is defined by hexadecimal number, for instance 0xFF means 255 that makes L_m to be approximately 11 m. Finally the number 43 is defined by the code inherent in the sensor.

$$L_m(mm) = (reg \cdot 43) + 43 \tag{4}$$

T-test

To examine the significant of both the methods, t-test was employed. There are two criteria in the t-test. Firstly H_o, which means there is no significant in the both methods. Secondly H_a, is the opposite of the aforementioned criteria. The methods are included into H_o criteria if in equation(5) is fulfilled. Otherwise the methods belong to H_a criteria. t calculation formula follows equation(6), and t table is defined in the t-test table. T calculation is the function of the mean of the data, the amount of the data, and also variant of both the methods.

$$t_{table}^{-} \leq t_{calc} \leq t_{table}^{+} \quad (5)$$

$$t_{calc} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \quad (6)$$

Experimental Setup

In the experiment, main board was connected with two external hardware, ultrasonic sensor and servo motor. Whereas LCD and LED, had been initially attached to the main board. The servo motor which rack and pinion mechanisms are installed on was operated by different main board to keep RTOS and primitive interrupt experiment comparable. The installation of the system is shown in Figure 5. In Figure 5. number 1 is rack and pinion mechanism, number 2 is ultrasonic sensor, number 3 is servo motor, number 4 is LED, and number 5 is LCD.

Figure 6 shows the setup of the sensor. While rack was reciprocally moving from left to right and vice versa, ultrasonic sensor transmitted the wave in certain period. There are two positions of which the ultrasonic sensor is imposed to detect the object, the starting point (x cm) and the end point (10 cm). If the rack was in the prescribed position, the ultrasonic sensor would detect the rack and deemed it as hit. Otherwise the sensor would mull it as miss. Every hit was counted and accumulated, then eventually would be recorded and compared to the real hit for both the methods. Furthermore, there were three main scenarios conducted in the reaserch. Those scenarios were based on the period of the rack mechanism (Table 5). The period of the rack were stipulated by altering the angular movement of the servo motor which rotated the rack mechanism. Larger the angular movement, longer the period would be. Moreover, to diminished the period, the loop number in the code was reduced so that the delay in the edge point (turning point) would me decreased. Hence the rack period would be declined.

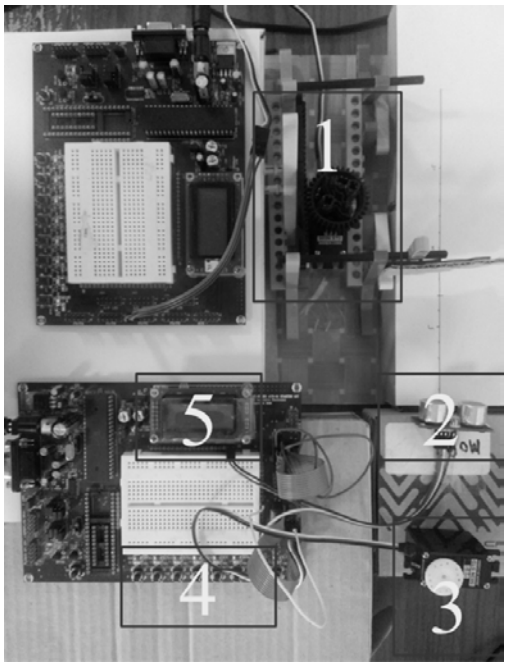


Figure. 5. Installation of the system.

The success rate of the ultrasonic sensor detect the occurrence of rack was used as performance indicator of both the methods. The observing time of the system were prescribed and altered after several iterations. The observation time wass ranging from 10 s to 50 s with the increment of 10 s. The amount of success reading by sensor are compared with the true value of the counting which is acquired by manual calculation and measurements. Parameters employed in this research are presented in Table 6. Figure 7 illustrates the schedulling of the tasks in RTOS. There are four tasks connected to the RTOS scheduller. Each task connection has two direction of operation, from and to RTOS scheduller. Meanwhile, the sensor task is also connected to LCD to display the data obtained by the sensor.

In non-preemptive interrupt, three main codes for determining the behavior of the interrups are represented in Figure 8, Figure 9, and Figure 10. In Figure 8. `ISR` is declared and empower `TIMER0_COMP` vector to stipulate the task's priority. The interrupt timer was set to be 1 ms, therefore the Output Compare Register (`OCR0`) was defined to be 187.5, the value of which is elaborated in the Figure 7. The task delay is explained in the three last rows of the code. The counting variable was initially set to be 2000 in this research. There-upon, in every rising clock which was prescribed to be 1 ms the counting value would be one value lower than the prior value. When the counting value reach 0 the task would be conducted which means the delay for each task is 2 s. The detail of which is elucidated in Figure 10.

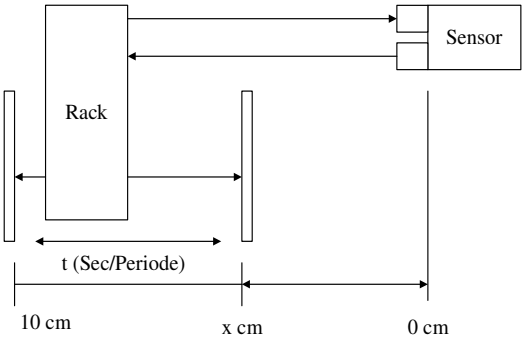


Figure 6. Scheme of the data acquisition.

TABLE 5
SCENARIO OF THE EXPERIMENT

Scenario	x (cm)	t (s)
1	7	1.7
2	7	2.6
3	5	3

Figure 9 shows the enable control of the interrupt. Corresponding to the desired period of the timer (1 ms) and CPU (Central Processing Unit) speed (12 MHz), the clock needed to attain it is 12000 clocks. Thereby the clock would rise after 12000 times of the CPU periodically rose. Interrupt mode which is used in this study is CTC (Clear Time on Compare), and the prescaler used is 64. Because of that, the TOP value of CTC becomes 187.5 as the result of the division of 64 on 12000. Hence, as written in the previous paragraph the value of `OCR0` is set to be 187 to approximate the 187.5 value. Finally, in Figure 7 `sei()` macro is declared which means the declaration of the global interrupt activation.

In Figure 10 tasks are called in acccordance to the delay. The delay is determined by the counting variable. When the counting variable less than or equal 0, the task will be undertaken, and vice versa.

3. Results and Analysis

There are three main results in the experiment which was conducted. First of all is hit rate of the system for both RTOS and interrupt method (shown is Figure 11, Figure 14, and Figure 17). Secondly, the relative error of both the methods by means of the comparison with the real hit which are illustrated in Figure 12, Figure 15, and Figure 18 for the three scenarios. Finally, Figure 13, Figure 16, and Figure 19 represent the standard deviation of the data obtained from the experiment.

According to Figure 11 RTOS hit rates in 1.7 rack period were closer to the real hit rates than interrupt hit rates for both the turning points (start and end). This means RTOS accuracy was better

TABLE 6
PARAMETERS USED IN EXPERIMENT

Symbol	Parameter	Value
V	Servo motor voltage (V)	4.8
n	Servo motor speed (rpm)	43.47
i	Number of iterations	10

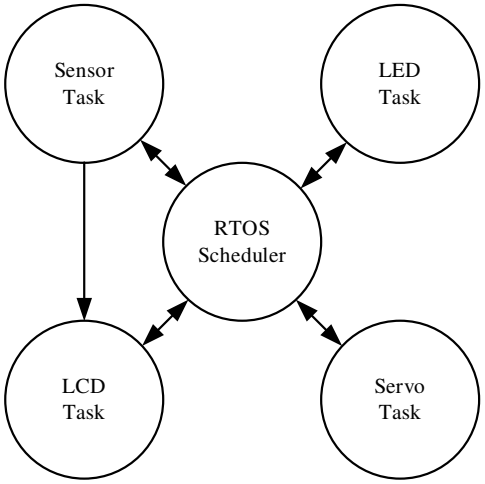


Figure 7. Scheme of task schedulling.

```
ISR(TIMER0_COMP_vect) {
    time++;
    if  (OCR0 == 187)  OCR0 = 188;
    else OCR0 = 187;
    for (int i = 0; i < task_count(); ++i)
{
        if (all_task[i].counting !=
        INT16_MIN && all_task[i].counting
        != INT16_MAX) {
            all_task[i].counting--;
                }
    }
}
```

Figure 8. ISR declaration in non-preemptive interrupt.

than interrupt accuracy. The aforementioned deduction was strengthened by Figure 12 the relative errors of the interrupt method were higher than RTOS methods. The errors of RTOS method fluctuated near 40% for starting point and 60% for ending point. Whereas the errors of interrupt were approximately 85% for starting point and 90% for ending point. On the contrary, as shown in Figure 13 RTOS results varied wider than interrupt method.

RTOS standard deviation values were higher than interrupts standard deviation, above all while the observation time was longer for both the turning points. For instance, the standard deviation of RTOS in 50 s observation time was almost thrice higher than interrupt method for starting point and 50% higher for ending point. These results indicated that interrupt method was more stable than

```
// CTC mode is used for interrupt setup
// CPU speed is 12 Mhz, hence 12000 clock
is needed to acquire 1 ms
// prescaler is set to be 64, thus TOP for
CTC is 12000/64 = 187.5
OCR0 = 187;
TCNT0 = 0;
TCCR0|=_BV(WGM01)|_BV(CS01)|_BV(CS00);
TIMSK |= _BV(OCIE0);
sei();
```

Figure 9. Interrupt enable control in non-preemptive interrupt.

```
for (int i = 0; i < task_count(); ++i)
{
    if (tmp[i] <= 0) {
    tmp[i] = all_task [i].function();
    if (tmp[i] < 0) tmp [i] = 0;
    }
    else
    {
       tmp [i] = 0;
    }
}
```

Figure 10. Task control in non-preemptive interrupt.

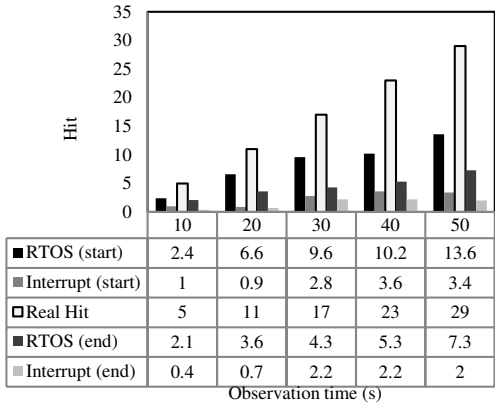
RTOS method. This means interrupt method was more precise than RTOS method, even though less accurate.

The trend for another scenarios are quite similar. The accuracy of the RTOS was higher than the accuracy of interrupt, and interrupt was more precise than RTOS depicted in the smaller standard deviation.

Meanwhile, the comparison based on the scenario in RTOS does not show the propensity of the results. The average relative errors for 2.6 s and 3 s in starting point are about 10% and 20% respectively, whereas in the ending point are around 40% for both the scenarios. In contrast, in interrupt results show the tendency of the decreasing error as the rack period increased. The average relative error for 2.6 s and 3 s period is starting poin are approximately 80% and 60% consecutively, whereas in the ending point are near the value of 80% and 70% respectively.

Finally, in the standard deviation figure observation time does not give a certain trendline by means that each scenario has its own trendline. Firstly in 1.7 s rack period scenario for RTOS method, the standard deviation rise as the observation time increase in the starting point, whereas in the interrupt, the value rise until reach the culmination point (30 s) then drop and stable.

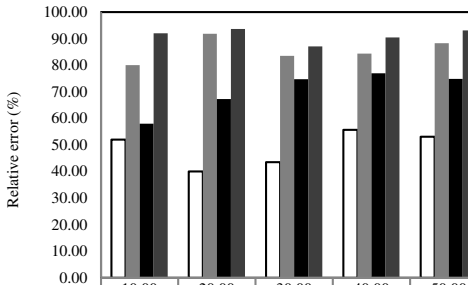
On the other hand, the standard deviation value in the 1.7 s scenario for both the methods remain stable throughout the observation time in ending point. Meanwhile, in the other two scenarios the standard deviation value fluctuate irregularly for each sub-scenario. According to the experiment conducted, RTOS exhibited higher accuracy than interrupt method, but surprisingly interrupt method has higher precision than RTOS. There are several reasons for the aforementioned occurring results.



	10	20	30	40	50
RTOS (start)	2.4	6.6	9.6	10.2	13.6
Interrupt (start)	1	0.9	2.8	3.6	3.4
Real Hit	5	11	17	23	29
RTOS (end)	2.1	3.6	4.3	5.3	7.3
Interrupt (end)	0.4	0.7	2.2	2.2	2

Observation time (s)

Figure 11. Average hit rate of RTOS, Interrupt, and manual calculation for 1.7 s period.



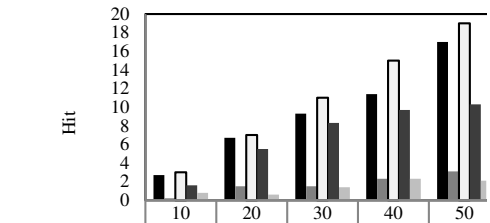
	10.00	20.00	30.00	40.00	50.00
RTOS (start)	52.00	40.00	43.53	55.65	53.10
Interrupt (start)	80.00	91.82	83.53	84.35	88.28
RTOS (end)	58.00	67.27	74.71	76.96	74.83
Interrupt (end)	92.00	93.64	87.06	90.43	93.10

Observation time (s)

Figure 12. Relative error of RTOS and Interrupt for 1.7 s period.

First of all, concerning to the accuracy result RTOS exhibited better results due to its schedulling technique. While one task is in the running state, the other tasks are stored in the ready state. This means when the running task is over, the other task from the ready state could be delivered to the running state directly. Because of it, the exchanging process of the tasks would be faster. If the exchanging process of the tasks is faster it would lead to the more frequent of the ultrasonic wave excitation. The more ultrasonic wave is shot, the higher the possibility of rack detection.

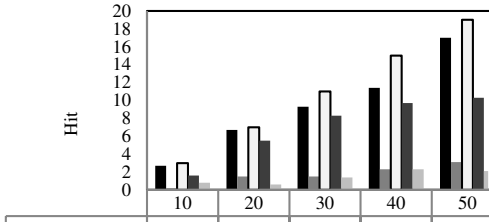
In contrast, interrupt method consider interrupt as a task. Therefore, there would be delays every time interrupt was called. This trait would decelerate the exchanging process of the tasks. Moreover, the task in the interrupt method would be executed sequantially by means while a certain



	10	20	30	40	50
RTOS (start)	2.7	6.7	9.3	11.4	17
Interrupt (start)	0.2	1.5	1.5	2.3	3.1
Real Hit	3	7	11	15	19
RTOS (end)	1.6	5.5	8.3	9.7	10.3
Interrupt (end)	0.8	0.6	1.4	2.3	2.1

Observation time (s)

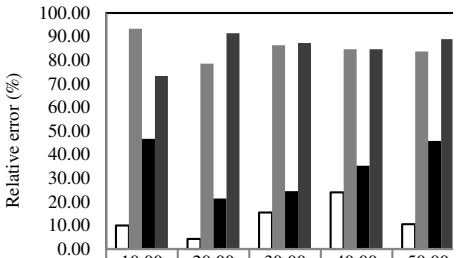
Figure 14. Average hit rate of RTOS, Interrupt, and manual calculation for 2.6 s period.



	10	20	30	40	50
RTOS (start)	2.7	6.7	9.3	11.4	17
Interrupt (start)	0.2	1.5	1.5	2.3	3.1
Real Hit	3	7	11	15	19
RTOS (end)	1.6	5.5	8.3	9.7	10.3
Interrupt (end)	0.8	0.6	1.4	2.3	2.1

Observation time (s)

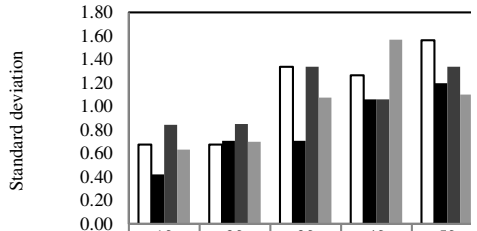
Figure 14. Average hit rate of RTOS, Interrupt, and manual calculation for 2.6 s period.



	10.00	20.00	30.00	40.00	50.00
RTOS (start)	10.00	4.29	15.45	24.00	10.53
Interrupt (start)	93.33	78.57	86.36	84.67	83.68
RTOS (end)	46.67	21.43	24.55	35.33	45.79
Interrupt (end)	73.33	91.43	87.27	84.67	88.95

Observation time (s)

Figure 15. Relative error of RTOS and Interrupt for 2.6 s period.



	10	20	30	40	50
RTOS (start)	0.67	0.67	1.34	1.26	1.56
Interrupt (start)	0.42	0.71	0.71	1.06	1.20
RTOS (end)	0.84	0.85	1.34	1.06	1.34
Interrupt (end)	0.63	0.70	1.07	1.57	1.10

Observation time (s)

Figure 16. Standard deviation of RTOS and Interrupt for 2.6s period.

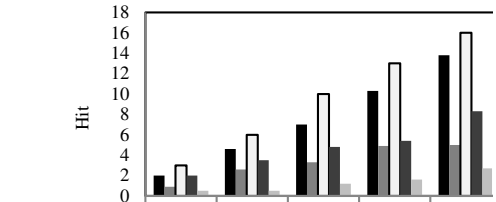
task was running, the other task could not interfere it. The aforementioned deduction was proved by the behavior of the system during experiment. Even though the delay of the LED was prescribed by 1 s, but in the experiment the LED delay was more than 1 s, more precisely the LED altered after the servo motor finished its task.

Secondly, regarding the precision of the system interrupt have better performance on repetition. The results variation of the interrupt was narrower than RTOS. The reason of which is because of the task undertaking process. In the interrupt the consecutive tasks were undertaken repeatedly with the similar sequence and with the same period. This behavior made the time when the ultrasonic wave was transmitted similar each period. Hence, the hit rate of the system using interrupt would not vastly varied.

On the other hand, RTOS task executing process varied each period. It is shown in the experiment that the LED task complied with the code (1s) delay. This means that there were alternation when the exchanging process of the task occured. Because of it, the sequence of the task execution was differrent each period. Thus, the time of when the sensor transmitted the ultrasonic wave was a time arbitrary task. This lead to the widely variation of the data acquired from the RTOS method.

In RTOS rack period did not affect the accuracy of the ultrasonic sensor because the delay of the sensor was only from the interval time of the ultrasonic sensor. On the contrary in the interrupt, the longer rack period time higher the accuracy this due to the delay which existed in the interupt was not only from the sensor but also from the interrupt itself. Thus, longer the period higher the probability of the detection would be. Finally in the standard deviation, neither observation time nor rack period gave certain behavior to the system since there is no solid connectivity between the schedulling and either observation time or rack period.

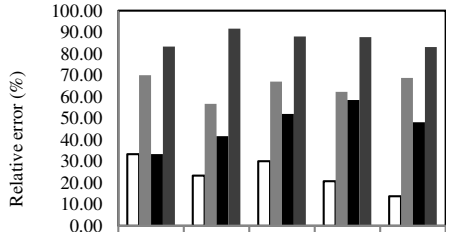
After the data have been entirely obtained, t-test was conducted to examine the significant of



	10	20	30	40	50
RTOS (start)	2	4.6	7	10.3	13.8
Interrupt (start)	0.9	2.6	3.3	4.9	5
Real Hit	3	6	10	13	16
RTOS (end)	2	3.5	4.8	5.4	8.3
Interrupt (end)	0.5	0.5	1.2	1.6	2.7

Observation time (s)

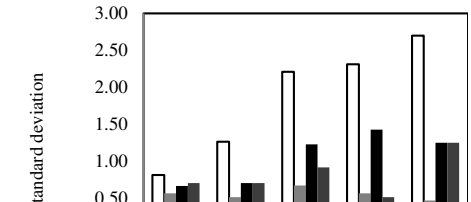
Figure 17. Average hit rate of RTOS, Interrupt, and manual calculation for 3s period.



	10.00	20.00	30.00	40.00	50.00
RTOS (start)	33.33	23.33	30.00	20.77	13.75
Interrupt (start)	70.00	56.67	67.00	62.31	68.75
RTOS (end)	33.33	41.67	52.00	58.46	48.13
Interrupt (end)	83.33	91.67	88.00	87.69	83.13

Observation time (s)

Figure 18. Relative error of RTOS and Interrupt for 3 s period.



	10	20	30	40	50
RTOS (start)	0.82	1.26	2.21	2.31	2.70
Interrupt (start)	0.57	0.52	0.67	0.57	0.47
RTOS (end)	0.67	0.71	1.23	1.43	1.25
Interrupt (end)	0.71	0.71	0.92	0.52	1.25

Observation time (s)

Figure 19. Standard deviation of RTOS and Interrupt for 3s period.

TABLE 7
T -CALCULATION OF EACH SCENARIO

Group	Rack Period (s)		
	1.7	2.6	3
Start (10 s)	11.03	18.75	12.39
Start (20 s)	22.68	38.34	14.04
Start (30 s)	12.56	18.39	6.63
Start (40 s)	15.29	22.17	8.98
Start (50 s)	3.13	22.43	8.15
End (10 s)	18.90	9.30	11.82
End (20 s)	5.88	22.12	17.89
End (30 s)	3.12	14.40	10.96
End (40 s)	10.87	14.06	10.74
End (50 s)	10.42	18.15	15.70

the methods (Table 7). The t-test shows that RTOS and interrupt are significant because the t_{calc} value more than t_{table} (2.101) for each scenario. This means employing RTOS method will significantly differ with utilizing interrupt method.

4. Conclusions

To recapitulate, RTOS has given a higher accuracy performance about 40 % than interrupt in every starting point scenarios and about 20 % in every end point scenarios due to the higher probability of the sensor detects the rack because of it is more frequent ultrasonic wave transmission. On the contrary, interrupt method exhibited narrrower output data variance (higher precision) compared to the RTOS, interrupt has given a smaller standard deviation than RTOS in all scenarios. Example, in scenario's with period 1.7 s, interrupt standard deviation is about 1.03 (starting point) and 1.14 (end point) against RTOS standard deviation is about 1.71 (starting point) and 2.54 (end point), then with period 2.6 s, interrupt standard deviation is about 0.71 (starting point) and 1.07 (end point) against RTOS standard deviation is about 1.34 (starting point) and 1.34 (end point) and finally, with period 3 s, interrupt standard deviation is about 0.67 (starting point) and 0.92 (end point) against RTOS standard deviation is about 2.21 (starting point) and 1.23 (end point). The reason is, the interrupt method with sequential processs did not vary at each period, unlike in the RTOS method. Thus, while choosing the perfect machine for industry, optimizations are needed to be conducted which is restricted by the output tolerance of the product itself. Therefore a better choice of either RTOS or interrupt is driven by the industry's product specification.

References

[1] Ş. Özgürler, A.F. Güneri, B. dır Gülsün, & O. Yılmaz, "Robot Selection for a Flexible Manufacturing System with AHP and TOPSIS Methods," *In 15th International Research/Expert Conference,* pp. 333-336, 2011.

[2] R. Kumar & R.K. Garg, "Optimal Selection of Robots by Using Distance Based Approach

Method", *Robotics and Computer-Integrated Manufacturing*, vol. 26, pp. 500-506, 2010.

[3] S.L. Tan, & T.N.B. Anh, "Real-time operating system (RTOS) for small (16-bit) Micro-controller" *In Proceeding of The 13th IEEE International Symposium on Consumer Electronics*, pp. 1007-1011, 2009.

[4] J. Lee, V.J. Mooney, A. Daleby, K. Ingstrom, T. Klevin & L. Lindh, "A Comparison of the RTU Hardware RTOS with a Hardware/Software RTOS" *In Proceeding of IEEE*, pp. 683-688, 2003.

[5] M.H Neishaburi, M. Daneshtalab, M.R. Kakoee, S. Safari, "Improving Robustness of Real-Time Operating Systems (RTOS) Services Related to Soft-Errors" *In Proceeding of IEEE*, pp. 528-534, 2007.

[6] N. Maruyama, T. Ishihara, H. Yasuura, "An RTOS in Hardware for Energy Efficient Software-based TCP/IP Processing" *In Proceeding of 8thIEEE Symposium on Application Specific Processor (SASP)*, pp. 58-63, 2010.

[7] L. Xianhua, Y. Kui, W. Wei, & Z. Wei, "A Control Architecture of Autonomous Robot and its Realization using Multiple DSP devices and RTOS Based Platform" *In Proceeding of The IEEE International Conference on Robotics, Intelligent Systems, and Signal Processing*, pp. 519-523, 2003.

[8] M.O.F. Sarker, C.H. Kim, J.S. Cho, & B.J. You, "Development of a Network-based Real-Time Robot Control System over IEEE 1394: Using Open Source Software Platform" *In Proceeding of The 3rd IEEE International Conference on Mechatronics (ICM)*, pp. 563-568, 2006.

[9] W. Atmadja, B. Christian, L. Kristofel, "Real Time Operating System on Embedded Linux with Ultrasonic Sensor for Mobile Robot" *In Proceeding of IEEE International Conference on Industrial Automation and Information & Comunication Technology (IAICT)*, pp. 22-25, 2014.

[10] H. Wei, Z. Huang, Q. Yu, M. Liu, Y. Guan, & J. Tan, "RGMP-ROS: a Real-time ROS Architecture of Hybrid RTOS and GPOS on Multi-core Processor" *In Proceeding of The IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2482-2487, 2014.

[11] Real Time Engineers Ltd, FreeRTOS, http://www.freertos.org., retrieved March 3, 2015.

[12] Real Time Engineers Ltd, FreeRTOS, http://www.freertos.org/RTOS-task-priority.html, retrieved March 3, 2015.

[13] Atmel-8155D-AVR-Atmega32A-Datasheet, February 2014.